在UML系统开发中有三个主要的模型：   
  
　　功能模型: 从用户的角度展示系统的功能，包括用例图。   
　　对象模型: 采用对象，属性，操作，关联等概念展示系统的结构和基础，包括[类图](http://www.baike.com/wiki/%E7%B1%BB%E5%9B%BE" \o "类图" \t "http://www.cnblogs.com/Yogurshine/archive/2013/01/14/_blank)、对象图、包图。   
　　动态模型: 展现系统的内部行为。 包括序列图，活动图，状态图。

UML的重要内容可以由以下五种类图定义

**用例图**：从用户角度描述系统功能，并指各功能的操作者。

**静态图**：包括类图，包图，对象图。

     类图：描述系统中类的静态结构

     包图：是包和类组成的，表示包与包之间的关系，包图描述系统的分层结构

     对象图：是类图的实例

**行为图**：描述系统动态模型和对象组成的交换关系。包括状态图和活动图

     活动图：描述了业务实现用例的工作流程

     状态图：是描述状态到状态控制流，常用于动态特性建模

**交互图：**描述对象之间的交互关系

   顺序图：对象之间的动态合作关系，强调对象发送消息的顺序，同时显示对象之间的交互

   合作图：描述对象之间的协助关系

**实现图：**

   配置图：定义系统中软硬件的物理体系结构

UML包括用例图、类图、构件图、部署图、顺序图、协作图、状态图、活动图

各个图简介：

用例图从外部用户的角度捕获系统的行为。他将系统功能划分为对活动者具有意义的事务。

类图描述类、接口、写作以及它们之间的关系的图

顺序图描述了随时间安排的一系列消息。每个分类角色显示为一条生命线，代表整个交互期间的角色。消息则显示为生命线之间的箭头。

协作图对交互中存在意义的对象和链建模

状态图用来描述一个特定对象所有可能的状态 以及由于各种事件的发 生而引起的状态之间的转移和变化。

用例图主要用来描述“用户、需求、系统功能单元”之间的关系。它展示了一个外部用户能够观察到的系统功能模型图。

　　【用途】：帮助开发团队以一种可视化的方式理解系统的功能需求。

　　用例图所包含的元素如下：

**1.** **参与者(Actor)**

　　表示与您的应用程序或系统进行交互的用户、组织或外部系统。用一个小人表示。



**2.** **用例(Use Case)**

　　用例就是外部可见的系统功能，对系统提供的服务进行描述。用椭圆表示。



**3. 子系统(Subsystem)**

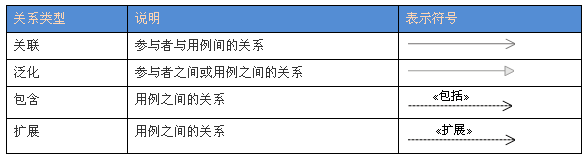
　　用来展示系统的一部分功能，这部分功能联系紧密。



**4.** **关系**

　　用例图中涉及的关系有：关联、泛化、包含、扩展。

　　如下表所示：



**a. 关联(Association)**

　　表示参与者与用例之间的通信，任何一方都可发送或接受消息。

　　【箭头指向】：指向消息接收方



**b. 泛化(Inheritance)**

　　就是通常理解的继承关系，子用例和父用例相似，但表现出更特别的行为；子用例将继承父用例的所有结构、行为和关系。子用例可以使用父用例的一段行为，也可以重载它。父用例通常是抽象的。

　　【箭头指向】：指向父用例



**c. 包含(Include)**

包含关系用来把一个较复杂用例所表示的功能分解成较小的步骤。

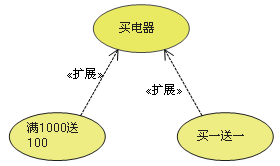
　　【箭头指向】：指向分解出来的功能用例



**d. 扩展(Extend)**

　　扩展关系是指用例功能的延伸，相当于为基础用例提供一个附加功能。

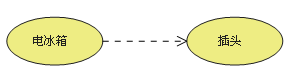
　　【箭头指向】：指向基础用例



**e. 依赖(Dependency)**

　　以上4种关系，是UML定义的标准关系。但VS2010的用例模型图中，添加了依赖关系，用带箭头的虚线表示，表示源用例依赖于目标用例。

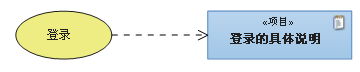
　　【箭头指向】：指向被依赖项

****

**5. 项目(Artifact)**

　　用例图虽然是用来帮助人们形象地理解功能需求，但却没多少人能够通看懂它。很多时候跟用户交流甚至用Excel都比用例图强，VS2010中引入了“项目”这样一个元素，以便让开发人员能够在用例图中链接一个普通文档。

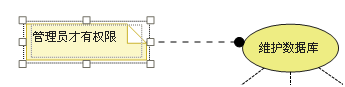
　　用依赖关系把某个用例依赖到项目上：



　　然后把项目-》属性 的Hyperlink设置到你的文档上；

　　这样当你在用例图上双击项目时，就会打开相关联的文档。

**6. 注释(Comment)**



**包含(include)、扩展(extend)、泛化(Inheritance)** **的区别：**

　　条件性：泛化中的子用例和include中的被包含的用例会无条件发生，而extend中的延伸用例的发生是有条件的；

　　直接性：泛化中的子用例和extend中的延伸用例为参与者提供直接服务，而include中被包含的用例为参与者提供间接服务。

　　对extend而言，延伸用例并不包含基础用例的内容，基础用例也不包含延伸用例的内容。

　　对Inheritance而言，子用例包含基础用例的所有内容及其和其他用例或参与者之间的关系；

**一个用例图示例：**

